*Institute for Applied Information Processing and Communications*

*Graz University of Technology*

IAIK

TUG

# Instruction Set Extensions for Fast Arithmetic in Finite Fields GF($p$) and GF($2^m$)

***Johann Großschädl and Erkay Savaş***

ECRYPT Meeting

Bochum, September 22, 2004

# Motivation

- **Elliptic curve cryptosystems**
  - Spend the majority of their execution time in a few performance-critical code sections (e.g. inner loops)
  - Speed up with Assembly code or ISA extensions
- **Instruction set extensions**
  - A set of custom instructions specifically designed to accelerate the processing of an application domain
  - Advantages: Performance, energy-efficiency, flexibility, scalability, cost-efficiency
  - Examples: SmartMIPS, ARM SecurCore, ST22 SmartJ

IAIK TUG

# Presentation Outline

- **Arithmetic in GF($p$)**
- **Arithmetic in GF($2^m$)**
- **MIPS32 architecture**
- **Proposed extensions to MIPS32**
  - Architectural enhancements
  - Custom instructions
- **Performance evaluation**

IAIK TUG

# ISA Extensions for ECC

- **Selection of arithmetic algorithms**
  - Several algorithms for the same operation
  - Multiple-precision multiplication: pencil-and-paper method, Comba's method, Karatsuba's method, ...
  - Algorithm exploration: Which algorithm to use?
- **Selection of custom instructions**
  - Huge number of candidate instructions
  - (Micro-)architectural constraints: instruction length, instruction format (3 operands), number of GPRs, ...
  - Compatibility to base architecture (MIPS32)

IAIK TUG

# Practical Relevance

- **Embedded systems**
  - Single application or application domain
  - Configurable and extensible RISC cores + tool chain (e.g. Tensilica Xtensa, ARC Tangent, etc.)
- **Multi-application smart cards**
  - 16 or 32-bit RISC processor (33 MHz)
  - Cryptographic co-processor not necessary for ECC
- **Future general-purpose processors**
  - Better support of cryptographic workloads
  - Advice to the computer architecture community

IAIK TUG

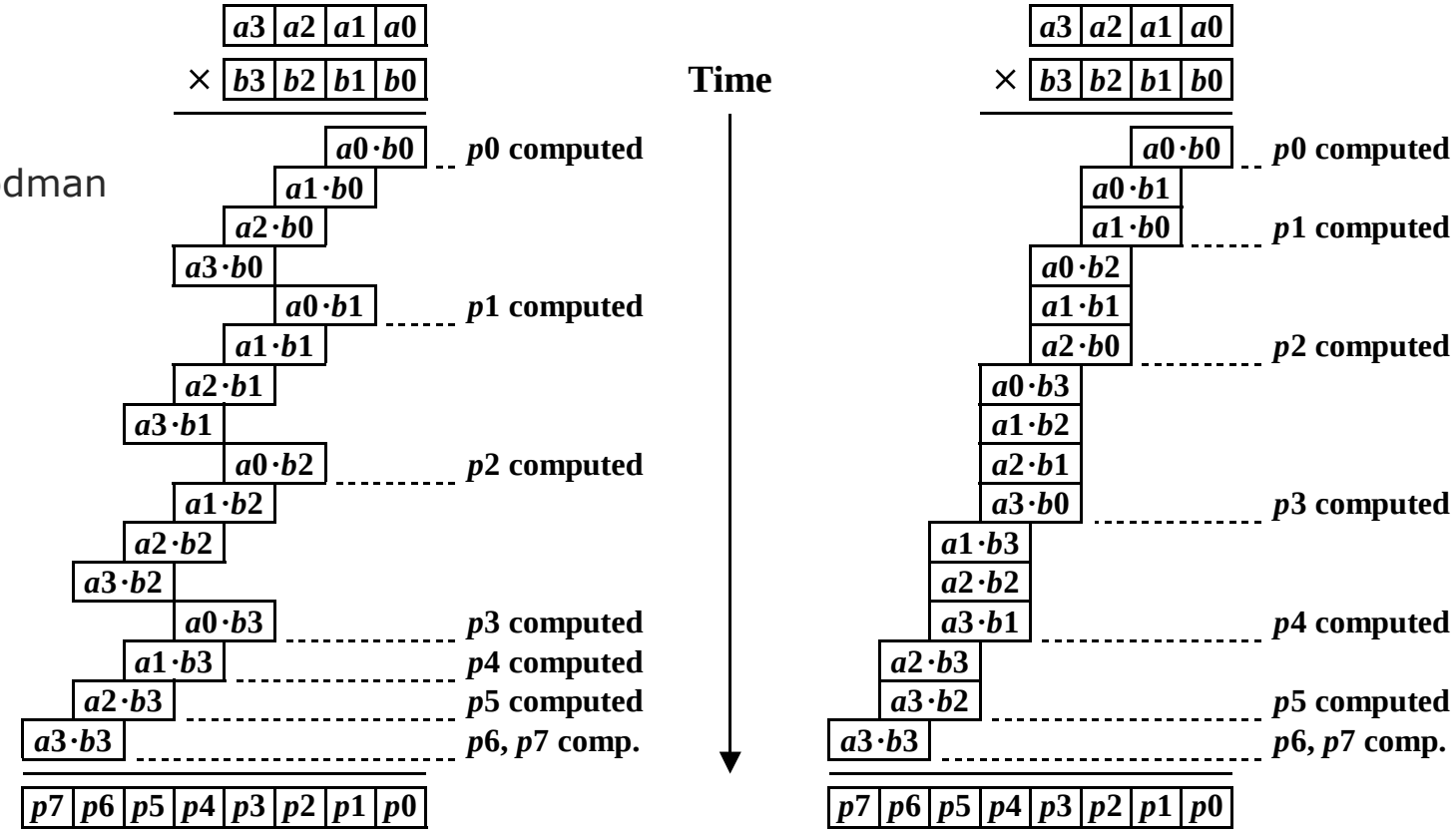
# Arithmetic in Prime Fields

- **Prime field GF($p$)**
  - Elements represented by arrays of $w$-bit words
  - Arithmetic: long integer mult. + modular reduction
- **Long integer multiplication**
  - Algorithms: pencil-and-paper, Comba, Karatsuba, ...
  - Realized with processor instructions (MUL, ADD, ...)
- **Modular reduction**
  - Generic algorithm: Montgomery (e.g. FIPS, FIOS)
  - Special primes allow much faster reduction, e.g. GM prime $p = 2^{192} - 2^{64} - 1$

IAIK TUG

# Multi-Precision Multiplication

© by
Jim Goodman




**Pencil-and-paper method** **Comba's method**

# Inner Loop Operations

- **Pencil-and-paper method**
  - Inner loop: $S \leftarrow a \times b + c + d$
  - $S$ is at most 64 bits when $a$, $b$, $c$, $d$ are 32 bits long
  - 2 LOADS and 1 STORE in inner loop
  - **UMAAL** instruction in ARMv6 architecture
- **Comba's method**
  - Inner loop: $S \leftarrow S + a \times b$ (multiply-accumulate)
  - Precision of $S$ exceeds 64 bits when a number of 64-bit products are added
  - 2 LOADS but no STORE in inner loop

# Arithmetic in Binary Fields

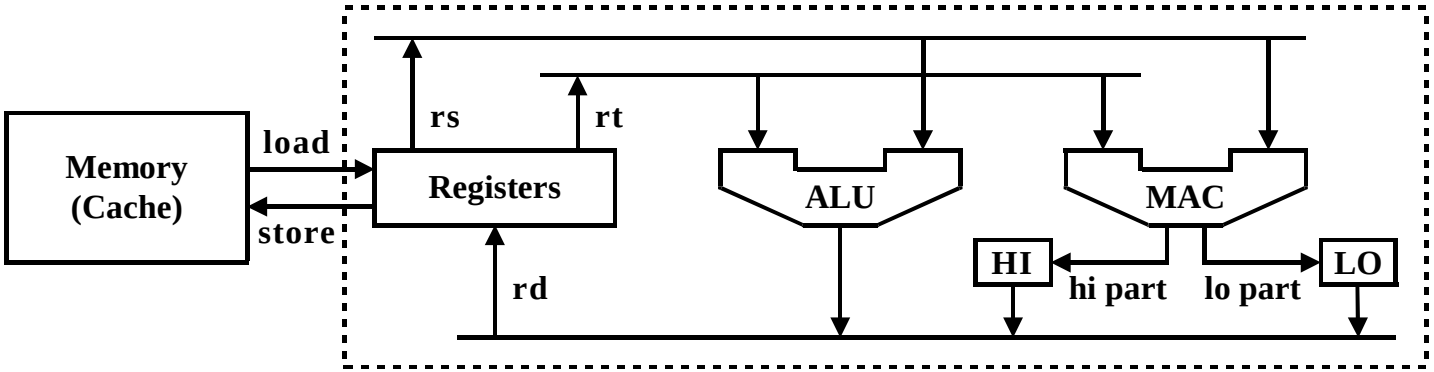
- **Binary extension field GF($2^m$)**
  - Elements are binary polynomials of degree $\leq m$
  - GF($2^3$) = { 0, 1, $t$, $t+1$, $t^2$, $t^2+1$, $t^2+t$, $t^2+t+1$ }
  - Addition is simple logical XOR
- **Multiplication of binary polynomials**
  - Basic algorithm: Shift-and-XOR
  - Faster: Left-to-right comb method (table look-up)
  - Squaring is much faster
  - Main drawback of GF($2^m$): polynomial arithmetic is not supported by general-purpose processors

IAIK TUG

# Faster Multiplication in GF($2^m$)

- **MULGF2 instruction**
  - Similar to the MUL instruction for integers, but interprets the operands as binary polynomials
- **Efficient arithmetic algorithms**
  - The availability of MULGF2 allows to use the same algorithms as in long integer arithmetic (Comba, ...)
- **Reduction modulo irreducible polynomial *p(t)***
  - Fast when $p(t)$ is a trinomial, e.g. $p(t) = t^{191} + t^9 + 1$ (requires only a few shift and XOR instructions)
  - Generic algorithm: Montgomery

# MIPS32 Architecture




- **Multiply instruction on MIPS32 4Km**
  - Single-cycle $(32 \times 16)$-bit multiply/accumulate unit
  - Separate pipeline, works in parallel with IU pipeline
  - Result accumulation registers: HI, LO (32 bits)
- **MADDU instruction**
  - **`MADDU rs, rt`** computes $(HI,LO) = (HI,LO) + rs \times rt$

# Proposed Enhancements

- **Wide accumulator**
  - 72-bit accumulator and 40-bit HI register
  - Sum up several 64-bit products without overflow
- **Unified multiplier**
  - Integers and binary polynomials use same datapath
  - Dual-field adders
- **Five custom instructions**
  - 3 for integer arithmetic, 2 for binary polynomials
  - All custom instructions are executed in the MAC unit
  - Simple to integrate into MIPS32

# Custom Instructions

| Format | Description | Operation |
|---|---|---|
| `MULTU rs, rt` | Multiply Unsigned | $(HI/LO) \leftarrow rs \times rt$ |
| `MADDU rs, rt` | Multiply and ADD Unsigned | $(HI/LO) \leftarrow (HI/LO) + rs \times rt$ |
| `M2ADDU rs, rt` | Multiply, Double and ADD Unsigned | $(HI/LO) \leftarrow (HI/LO) + 2rs \times rt$ |
| `ADDAU rs, rt` | ADD to Accumulator Unsigned | $(HI/LO) \leftarrow (HI/LO) + rs + rt$ |
| `SHA` | SHift Accumulator | $(HI/LO) \leftarrow (HI/LO) \gg 32$ |
| `MULGF2 rs, rt` | Multiply over GF(2) | $(HI/LO) \leftarrow rs \otimes rt$ |
| `MADDGF2 rs, rt` | Multiply and ADD over GF(2) | $(HI/LO) \leftarrow (HI/LO) \oplus rs \otimes rt$ |

- MULTU and MADDU are native MIPS32 instructions
- M2ADDU accelerates squaring
- ADDAU facilitates multi-prec. addition (GM reduction)
- MULGF2 / MADDGF2 are similar to MULTU / MADDU, but they interpret their operands as bin. polynomials

IAIK TUG

# Inner Loop of Comba's Method

```
label: LW    $t0, 0($t1)      # load A[j] into $t0
       LW    $t2, 0($t3)      # load B[i-j] into $t2
       ADDIU $t1, $t1, 4      # increment pointer $t1 by 4
       MADDU $t0, $t2         # (HI|LO)=(HI|LO)+($t0*$t2)
       BNE   $t3, $t4, label  # branch if $t3 != $t4
       ADDIU $t3, $t3, -4     # decrement pointer $t3 by 4
```

- Inner loop executes in 6 clock cycles
- MADDU does not occupy the write port of register file
- BNE instruction can execute during the latency period of the MADDU operation
- (32×32)-bit multiplier not necessary to reach peak performance (same perf. with (32×12)-bit multiplier)

IAIK TUG

# ECC Performance

| Arithmetic operation | GF($p$), $\|p\| = 192$ | GF($2^m$), $m = 191$ |
|---|---|---|
| Modular addition | 74 (155) | 62 |
| Comba multiplication w/o red. | 347 | 347 |
| Comba squaring w/o red. | 238 | 74 |
| Fast reduction (loop unrolled) | 65 | 75 |
| Montgomery multiplication | 594 (675) | 594 |
| Montgomery squaring | 447 (528) | 306 |
| Scalar multiplication (generic) | $1668 \cdot 10^3$ | $1040 \cdot 10^3$ |
| Scalar multiplication (optimized) | $1178 \cdot 10^3$ | $693 \cdot 10^3$ |

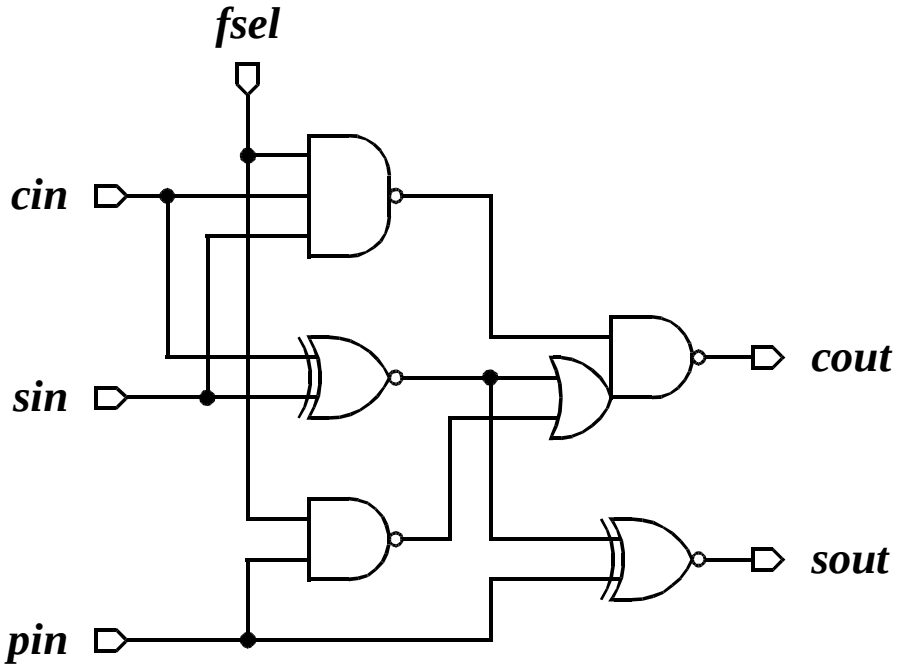
- Simulations with a cycle-accurate SystemC model
- Operations with fast reduction are 30-40% faster than generic algorithms with Montgomery reduction
- Speed-up to "ordinary" SW: GF($p$) 1.8x, GF($2^m$) 6x

# Unified INT/POLY Multiplier

A dual-field adder (DFA) is a full adder capable of performing addition both with and without carry

Integer mode: $fsel = 1$
Polynomial mode: $fsel = 0$

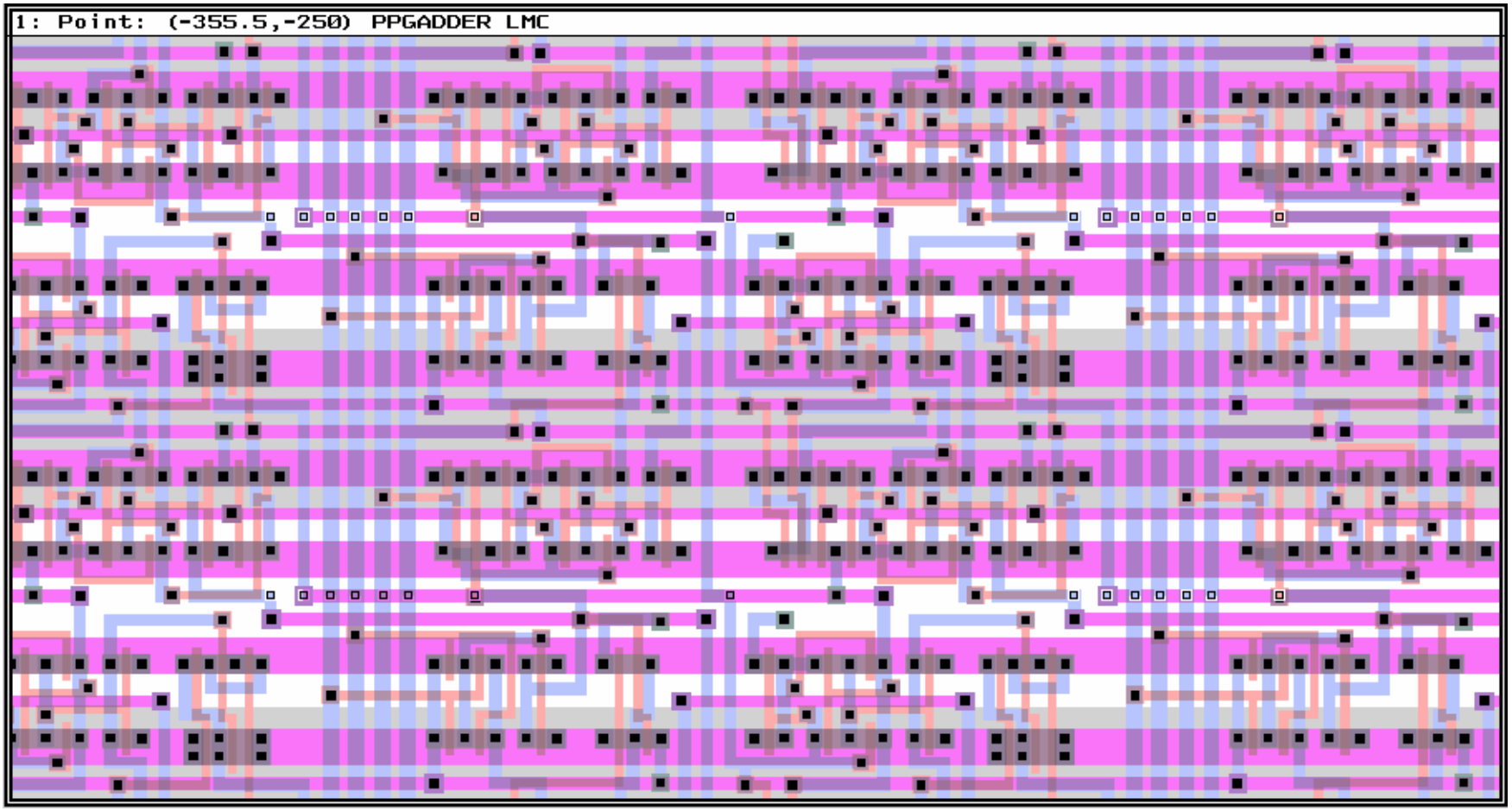
Only slightly larger than a conventional multiplier




A properly designed unified multiplier consumes 30% less power in POLY-mode than in INT-mode

IAIK TUG

# Low-Power Multiplier Datapath



IAIK TUG

# Concluding Remarks

- **Fast & flexible elliptic curve cryptography**
  - EC point mult. over 192-bit GF($p$) in 36 ms (33MHz)
  - EC point mult. over 191-bit GF($2^m$) in 21 ms
  - ISA extensions allow fast yet flexible ECC
- **Extensions simple to integrate into MIPS32**
  - Only five custom instructions allow to support a wide range of public-key systems (IF, DLP, ECDLP)
  - All custom instructions are executed in the MAC unit
  - Extra hardware cost is marginal (additional logic in the instruction decoder and the multiplier)

# Recent Results

- **FPGA prototype of a SPARC V8 core**
  - Based on the LEON2 processor (open source)
  - ISA extensions increase area by ~5% (core+cache)
  - Speed: 50 MHz (XC2V3000-4), >200 MHz (0.18 µm)
- **Domain-specific instruction set extensions**
  - Supports arithmetic in GF($p$), GF($2^m$), GF($3^m$), OEF
  - Energy-efficient software implementation
- **Example application: MatrixSSL with ECC**
  - Embedded SSL (70k executable for client + server)
  - System-on-chip: LEON + Ethernet + uCLinux + SSL

IAIK TUG

# Contact Info

Dipl.-Ing. Johann Großschädl
Graz University of Technology
Institute for Applied Information Processing & Communications
Inffeldgasse 16a, A-8010 Graz, Austria
Johann.Groszschaedl@iaik.at

Dr. Erkay Savaş
Sabanci University
Faculty of Engineering and Natural Sciences
Orhanli Tuzla, TR-34956 Instanbul, Turkey
erkays@sabanciuniv.edu

The first author was supported by the Austrian Science Fund (FWF) under grant number P16952-N04 ("Instruction Set Extensions for Public-Key Cryptography"). The second author was supported by The Scientific and Technical Research Council of Turkey under project number 104E007.